1. Pain Points in Using LLMs

- **Ambiguity:** The model sometimes gave vague or generic requirements without concrete details.
- **Inconsistency:** Asking the same question in slightly different ways sometimes led to contradictory requirements.
- **Overconfidence:** The model stated requirements confidently even when unnecessary or unrealistic.
- **Length of responses:** Some outputs were too long, requiring extra work to filter what was relevant.
- **Context limits:** The model occasionally “forgot” earlier parts of the conversation when the prompt chain got long.
- **Hallucination and accuracy:** Generated plausible-sounding but incorrect regulatory requirements or stakeholder needs.
- **Lack of domain-specific knowledge:** Generic responses missed nuanced food delivery industry specifics.
- **Inconsistent output formats:** Varying structure across prompts made consolidation difficult.
- **Over-generalization:** Tendency to produce generic use cases rather than actionable requirements.

---

2. Surprises

- Variability in responses: Slightly rephrased prompts could produce completely different sets of requirements, sometimes contradicting earlier outputs.
- Creative but non-standard requirements: The model suggested drone-based delivery or AI-powered meal recommendations which were interesting but not core requirements.
- Different focus depending on phrasing: Some prompts emphasized restaurant needs, others customer needs.
- Stakeholder bias revelation: The model sometimes highlighted unexpected biases (e.g., restaurant owners prioritizing profit over food safety).
- Regulatory complexity: The RAG system revealed intricate health/tax regulations.
- Edge case identification: Unusual use cases like accessibility needs, dietary restrictions, and fraud prevention were highlighted.
- Conflicting priorities: Different prompting approaches yielded contradictory stakeholder priorities.
- Open-ended, exploratory prompts: Forcing everything into a strict structure could reduce creativity or nuance, causing some content to be lost.
- Long or repeated input parameters: Repetition of long content across examples could overshadow other information.
- Reasoning in examples: Including reasoning in examples could distract and increase cognitive load.

---

3. What Worked Best

- **Structured prompts:** Listing functional, non-functional requirements, and constraints separately gave organized outputs.
- **Iterative refinement:** Breaking tasks into smaller steps (functional -> non-functional -> edge cases) improved results.
- **Role prompting:** Asking the model to act as a software analyst, product manager, or UX designer improved relevance.

- **Persona adoption:** “Act as a senior product manager” or similar prompts gave context.
- **Step-by-step decomposition:** Focus on one module at a time (e.g., user app, driver app, admin panel).
- **Specific formats:** Bullet lists, tables, JSON, or validation prompts helped structure outputs.

---

4. What Worked Worst

- **Open-ended questions:** “What requirements should a food delivery app have?” produced vague or repetitive outputs.
- **Overly complex prompts:** Packing too much into one query caused the model to ignore parts of the request.
- **Single-shot prompting:** One-time queries without iteration yielded incomplete requirements.
- **Unstructured consolidation:** Merging many use cases without a systematic approach was chaotic.
- **Lack of validation prompts:** Not asking the LLM to critique or validate outputs reduced reliability.
- **Single perspective:** Focusing on one stakeholder ignored other important viewpoints.

---

5. Pre-/Post-Processing

Pre-processing:
- Defining categories (functional, non-functional, user stories, constraints) for a clear framework.
- Providing context (e.g., “assume this is for a mid-sized city in India”).
- Structuring prompts consistently, including persona, objective, and explicit constraints.

Post-processing:
- Summarizing and consolidating overlapping points.
- Filtering unrealistic or irrelevant requirements (e.g., drones or AI chefs).
- Reformatting outputs into tables or requirement documents.
- Asking the LLM to group or summarize features for a concise, actionable overview.
- Manual review to remove redundancies and correct factual errors.

---

6. Best vs Worst Prompting Strategies

Best:
- Step-by-step decomposition of requirements.
- Asking for output in structured formats (bullet lists, tables, JSON).
- Role prompting and persona adoption.
- Focusing on one module at a time.
- Using validation prompts: “What might I have missed?” or “Critique this list.”

Worst:
- Very vague prompts (“Tell me everything about food delivery systems”).
- Overloading with too many tasks in one prompt.
- Asking for creativity when standardization was required.

Single perspective prompts ignoring other stakeholders.